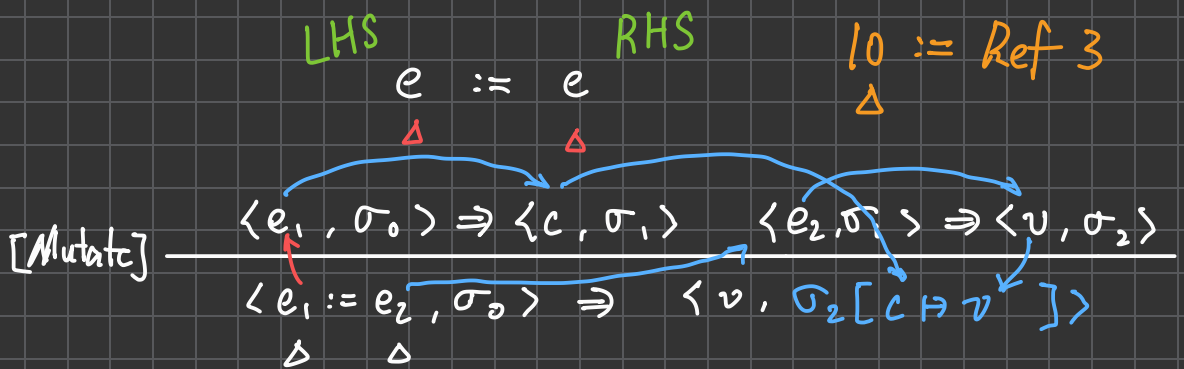


Let x = e_1 In e_2

C / C++

10 is not an l-value

```
int x = 3;          ✓   L-Value
10 = 5;             X   R-Value

int *ptr; // init
*ptr = 5;
char ** arr_of_strs; // initialized
arr_of_strs[3][6] = 'h';

arr[3] = arr[4];
LHS    RHS
```

F^b $\quad e \Rightarrow v$

F^b_S $\quad \langle e, \sigma_1 \rangle \Rightarrow \langle v, \sigma_2 \rangle$

Pure / Impure

↓

Languages w/ Side Effects

Exceptions ⊆ Side Effect

$F^b \rightarrow F^b X$ $\qquad e \Rightarrow v$

Py `def f():`

```
if error:
    return False
i = 0
j = i + 1
normal_comp()
return True
```

dead code

If error Then
  False
Else
  normal ... ;
  True

$F^b_{(;)}$ $\quad e_1 ; e_2 \quad (\text{Fun } x \rightarrow e_2)\ e_1$ where $x \notin FV(e_2)$

Py

```
while _ :
    if terminate :
        break
    else
        ...
continuation()
```

```
while _ :
    if skip : continue
```

Side-Effect : Affection Control Flow

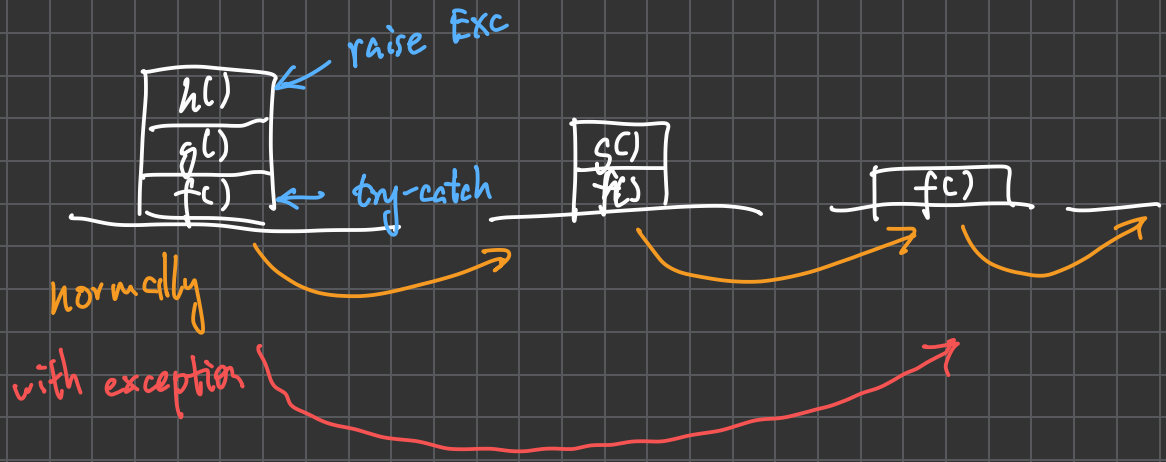
Exceptions : Call-Stack

Py

```
def f():
    try:
        g()
    except x:
        # handle x
```

```
def g():
    h()
def h()
    raise Exc()
```

f()



Side Effects

Effects affecting

- Memory
- Control flow
- Call-stack

IO / Input / output

- Network
- File · Op

Pure	$e \Rightarrow v$
Impure Mem	$\langle e, \sigma_0 \rangle \Rightarrow \langle v, \sigma_1 \rangle$
World	$\langle e, W \rangle \Rightarrow \langle e, W' \rangle$

Exception $F^b \to F^b X$ $\quad e \Rightarrow v$

$e ::= (\ldots F^b \text{ Exp } \ldots)$

$\mid$ Raise $\#l$ $\boxed{e}$ ← payload

Exception Name

$\mid$ try e_1 catch $\#l\ x \to e_2$

$v ::= (\ldots F^b \text{ Value } \ldots) \mid \text{Raise } \#l\ v$

$e ::= e_1 + e_2 \mid e_1 - e_2 \mid e_1 / e_2$

$e_1 / 0 \Rightarrow \text{Error}$

[Div by non-zero]
$$\frac{e_1 \Rightarrow n_1 \quad e_2 \Rightarrow n_2 \quad n_2 \neq 0}{e_1 / e_2 \Rightarrow n_1 / n_2}$$

[Div-by-zero]
$$\frac{e_2 \Rightarrow 0}{e_1 / e_2 \Rightarrow \text{Raise } \#\text{DivByZero } 0}$$

$$(\text{Fun } x \to \text{Fun } y \to 3 + x / y)\ 4\ 0$$

$$\rightsquigarrow \quad 3 + 4 / 0$$

$$\text{[Add!]}\ \frac{3 \Rightarrow 3 \qquad \text{[DBZ]}\ \dfrac{0 \Rightarrow 0}{4/0 \Rightarrow \text{Raise \#DivByZero } 0}}{3 + 4/0 \Rightarrow \text{Raise \#DivByZero } 0}$$

$$\text{[Add]}\ \frac{e_1 \Rightarrow n_1 \qquad e_2 \Rightarrow n_2 \qquad n_1, n_2 \in \mathbb{Z}}{e_1 + e_2 \Rightarrow n_1 + n_2}$$

$$\text{[Add-l-Err]}\ \frac{e_1 \Rightarrow \text{Raise \#}l\ p}{e_1 + e_2 \Rightarrow \text{Raise \#}l\ p}$$

$$\text{[Add-r-Err]}\ \frac{e_1 \Rightarrow n_1 \quad n_1 \in \mathbb{Z} \qquad e_2 \Rightarrow \text{Raise \#}l\ p}{e_1 + e_2 \Rightarrow \text{Raise \#}l\ p}$$

$$\text{[Add-l-TypeErr]}\ \frac{e_1 \Rightarrow v \qquad v \notin \mathbb{Z}}{e_1 + e_2 \Rightarrow \text{Raise \#TypeErr } v}$$

[Raise] $\dfrac{e \Rightarrow v}{\texttt{Raise \#}l\ e \Rightarrow \texttt{Raise \#}l\ v}$

[Double Raise] $\dfrac{e \Rightarrow \texttt{Raise \#}l\ v}{\texttt{Raise \#}l\ e \Rightarrow \texttt{Raise \#}l\ v}$

[Try Not Capturing] $\dfrac{e_1 \Rightarrow v \quad v \text{ is not } \texttt{Raise \#}l \ldots}{\texttt{try } e_1 \texttt{ catch \#}l\ x \to e_2 \Rightarrow v}$

[Try Fail] $\dfrac{e_1 \Rightarrow \texttt{Raise \#}l\ p}{\texttt{try } e_1 \texttt{ catch \#}l\ x \to e_2 \Rightarrow e_2[p/x]}$